Technická univerzita v Liberci

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: **B 2612 - Elektrotechnika a informatika**

Studijní obor: **2612R011 - Elektronické informační a řídící systémy**

Implementace IPv6 pro Linux

Linux IPv6 implementation

BAKALÁŘSKÁ PRÁCE

Autor práce: **Daniel Římal**

Vedoucí práce: **doc. RNDr. Pavel Satrapa, Ph.D.**

V Liberci 8.5.2007

ORIGINALNI ZADANI BP

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména §60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že souhlasím s případným užitím mé bakalářské práce (prodej, zapůjčení apod.)

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce.

V Liberci 8.5.2007

Poděkování

Zde bych rád poděkoval doc. RNDr. Pavlu Satrapovi, Ph.D. za podněty, připomínky, cenné rady a vedení při tvorbě této bakalářské práce.

Dále bych rád poděkoval svým rodičům a přítelkyni za podporu, trpělivost a důvěru při tvorbě bakalářské práce.

Anotace

Cílem bakalářské práce je seznámení s nastupujícím Internet Protokolem verze 6 (IPv6). V této práci jsou probrána jak pozitiva nové generace IP, tak i některé slabší stránky. Další část práce je náhled na vývoj a stávající podporu tohoto nastupujícího protokolu v linuxových operačních systémech a jejich aplikacích. Tato práce se také zaměřuje na porovnání přenosového výkonu IPv4 a IPv6.

Již poměrně dlouhá doba cíleného vývoje IPv6 stacku v Linuxu slibuje dobrou podporu IPv6 včetně odpovídající rychlosti. Taktéž uživatelské a serverové aplikace vesměs podporují IPv6. Rychlost IPv6 je srovnatelná se stávajícím protokolem. Z těchto dílčích závěrů vyplývá dobrá připravenost Linuxu na běžný provoz na IPv6.

Annotation

The intention of this bachelor thesis is familiarization with impending internet protocol version 6 (IPv6). In this thesis are examined new generation IP and also the otherweaknesses. Next part of this thesis is familiarization with development and allowance of this protocol in Linux and their application. This thesis also compare transmission rate IPv4 and IPv6.

A long time of development IPv6 stack in Linux promise good support of IPV6. Also user side and server side application consist good support of IPv6. A speed of IPv6 is comparable with IPv4. The result is: Linux is ready for using IPv6.

Obsah

Úvod

Výpočetní technika již několik let zaznamenává velký rozvoj, internet se rozšiřuje mílovými kroky a stále přibývá osobních počítačů, serverů, kapesních počítačů a dalších zařízení, která jsou schopná se připojit na internet, nejlépe odkudkoliv. A právě tady současný Internet Protokol ztrácí dech. Nejen že pomalu docházejí IPv4 adresy, ale také například bezpečnost a podpora mobilních zařízení není v zastaralém protokolu dostatečně rozvinutá.

V první kapitole budou zmíněny nedostatky současného IPv4 a vyzdvihnuty ty vlastnosti IPv6, které tyto problémy buď přímo nebo nepřímo řeší. Následný podrobnější pohled detailněji rozebírá zajímavé vlastnosti IPv6, jako je např. mobilita nebo autokonfigurace.

Připravenost operačních systémů na IPv6 je podrobněji popsána v druhé polovině práce. Konkrétně se jedná o operační systémy založené na jádře Linux. Mimo jiné je vzpomenuta kompatibilita s IPv6, možnosti připojení včetně praktických ukázek a také například rychlost implementace IPv6 v Linuxu.

Kapitola 1

Proč IPv6

1.1 Nedostatky IPv4

Kolem roku 1992 si organizace IETF (The Internet Engineering Task Force) začala uvědomovat počínající nedostatek adres internet protokolu. Existovaly statistické studie předpovídající vyčerpání IPv4 adres mezi lety 2005 a 2011. Aby tato věta nebyla zavádějící, je nutné zmínit, že důvodů vzniku nového protokolu bylo více, mezi nimi i některá technická omezení stávajícího starého protokolu. Přeci jen je vznik IPv4 datován rokem 1981.

Z tohoto důvodu roku 1994 vzniklo RFC 1752 označované jako "Požadavky protokolu nové generace". Z těchto počátečních požadavků vzniklo na konci roku 1995 po mnohých diskuzích RFC 1883, obsahující první návrh specifikace nového protokolu s názvem "Internet Protocol, Version 6 (IPv6) Specification", později revidovaného v RFC 2460.

1.2 Co přináší IPv6 nového

IPv6 vstupuje na trh coby evoluce protokolu IP verze 4. Přináší mnoho nových vlastností a některé stávající významně vylepšuje. Zde jsou uvedeny nejvýznamnější změny a vylepšení:

1. Významné rozšíření adresního prostoru

   Adresa IPv6 je rozšířena na 128 bitů oproti pouhým 32 bitům starší verze protokolu, což přináší obrovské množství adres. Pro představu - na jeden čtvereční metr zemské plochy připadá 665570793348866943898599 adres [1]. Tato změna, dá-li se to takto říci, je hnacím motorem přechodu na IPv6.

2. Autokonfigurační mechanismus

   Autokonfigurace je zřejmě jedna z nejúchvatnějších vlastností nového protokolu. V praxi by měla fungovat tak, že na síti přítomný router vysílá v pravidelných intervalech všechny náležitosti potřebné ke správnému samonakonfigurování, případně může počítač či obecněji zarízení připojené na síť o tyto informace samo požádat.

3. Zjednodušení hlaviček datagramu a jejich zřetězování

   IPv6 datagram na sobě nese podstatně jednodušší hlavičku než starší protokol IPv4. Je zde použita hlavička obsahující nejzákladnější údaje. Ostatní, volitelné údaje a podrobnosti nese datagram v dalších hlavičkách připojených k základní hlavičce. Tento mechanismu se nazývá zřetězování hlaviček.

IPv6 přináší podstatně více novinek a zlepšení. Podrobněji budou tyto finesy zmíněny dále.

1.3 IPv6 v praktickém životě

IPv6 běžného uživatele dosud nezasáhlo. Dá se říci, že ani člověk živící se moderní výpočetní technikou nemá o IPv6 mnoho povědomí.

Tato malá znalost a dosavadní nezájem o IPv6 je zřejmě způsobeno z velké míry fenoménem jménem NAT. Tento mechanismus překladu adres sice částečně řeší nedostatek adres, ale vzniká při tom množství dalších problémů, jako například nemožnost navázat přímé spojení s počítačem umístěným za routerem, na kterém běží NAT.

NAT je sice jakési bezpečnostní opatření, ale tyto přímo navazující spojení jsou v moderním internetu nezbytná. Používají se např. u IP telefonie, videokonferencí, ale problémem je např. i FTP spojení.

IPv6 tyto problémy poměrně úspěšně eliminuje. Přeze všechno je přístup odborné veřejnosti k novému protokolu doposavad poněkud liknavý.

Kapitola 2

Podrobnější pohled na IPv6

2.1 Struktura IPv6 paketu

Paket IP protokolů obou verzí začíná hlavičkou, za níž následují "užitečná" data. Oproti dosavadnímu protokolu doznala struktura základní hlavičky IPv6 výrazného zjednodušení. Několik položek ubylo a některé byly změněny.

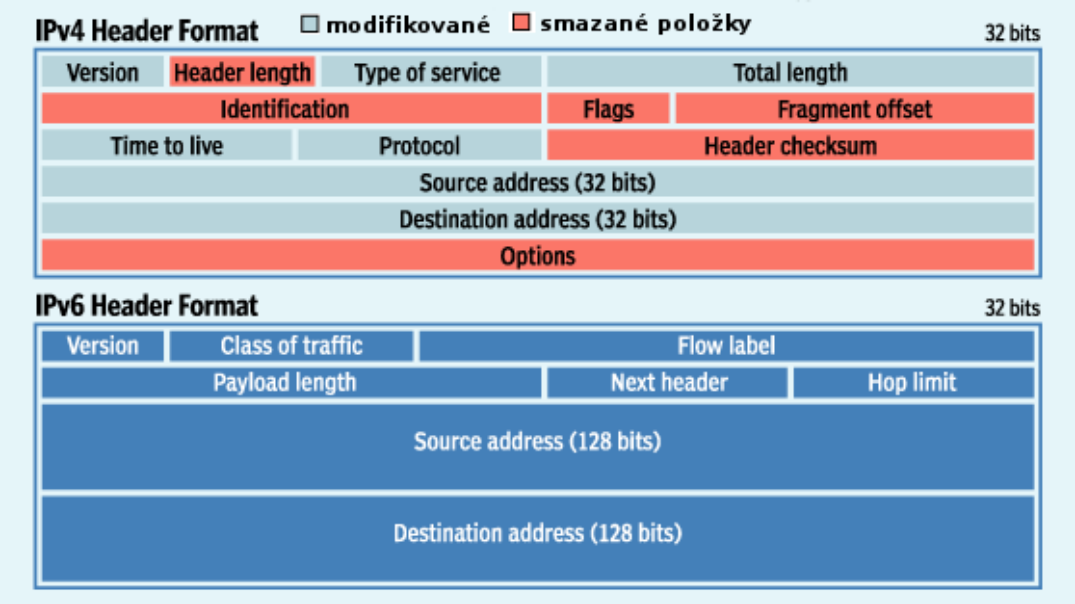
Následující výčet uvádí položky hlavičky, jenž byly oproti IPv4 hlavičce odebrány [3]:

1. Délka hlavičky
2. Identifikace, příznak, offset
3. Kontrolní součet hlavičky

Obrázek 2.1 srozumitelně ilustruje rozdíly hlaviček obou protokolů.

Pokud je zmíněna hlavička protokolu IPv6, mnohdy je u ní uvedeno slovo základní a to z důvodu revoluce protokolu IPv6 i v tomto směru. Nový způsob zacházení s hlavičkami je označován jako zřetězení hlaviček, případně rozšiřující hlavičky. Aktuální specifikace IPv6 definuje 6 rozšiřujících hlaviček [3], v původních názvech označované jako:

1. Hop-by-Hop Options header



Obrázek 2.1: IPv4 a IPv6 hlavičky[7]

2. Routing header

3. Fragment header

4. Destination Options header

5. Authentication header

6. Encrypted Security Payload header

Tyto rozšiřující hlavičky jsou připojovány k základní hlavičce a to z toho důvodu, je-li potřeba funkčnosti, jenž konkrétní hlavička zajišťuje. Toto zřetězení je uvedeno v základní hlavičce jako položka Next Header. Tutéž položku obsahuje každá další připojená hlavička která také obsahuje identifikátor následující připojené hlavičky nebo obsahuje identifikátor protokolu vyšší vrstvy, případně může obsahovat symbol označující tuto hlavičku za poslední (no next header) [3].

2.2 Adresování v IPv6

2.2.1 Poměry dosavadní a budoucí

32 bitů dnešní IPv4 adresy znamená přibližně 4.3 miliardy možných adres, přičemž je nelze použít všechny z důvodu nerozumného přidělování v ranných dobách sítí. Toto vše přispělo k nedostatku adres i přesto, že přístup k internetu má pouhých 14% obyvatel zeměkoule [4]. Potřeba adres bude stále narůstat. IP adresa již není pouze doménou počítačů, ale v moderním světě se vyskytuje i v mobilních telefonech, v lednicích obstarávajících nákupy, v autech starajících se o údržbu a mnohých dalších vymoženostech.

Jak již bylo zmíněno, nový protokol používá k adresování celých 128bitů. Nárůst adresovatelných zařízení ilustrujme na adresaci sítí a podsítí. V dobách adresních tříd IPv4 bylo možno adresovat 2 113 389 sítí. S nástupem beztřídního adresování (CIDR) toto číslo nepatrně vzrostlo [4].

Adresní prostor IPv6 globálních unicastových adres v současnosti umožňuje adresovat 35 184 372 088 832 sítí (prefix 48bitů), přičemž uvnitř každé může být rozlišeno dalších 65 536 podsítí [4].

2.2.2 Typy adres

IPv6 definuje 3 základní typy adres

1. Unicastová adresa

   Unicastová (individuální) adresa jednoznačně identifikuje síťové rozhraní.

2. Multicastová adresa

   Multicastová (skupinová) adresa identifikuje skupinu IPv6 zařízení, přičemž paket odeslaný na tuto adresu je doručen všem členům skupiny.

3. Anycastová adresa

Anycastová (výběrová) adresa je přiřazena více síťovým zařízením, přičemž paket odeslaný na tuto adresu je doručen pouze jednomu z nich, zpravidla tomu nejbližšímu.

2.2.3 IPv6 adresa

IPv6 adresa se zapisuje hexadecimálně do osmi bloků po 16 bitech, navzájem oddělených dvojtečkou [3]. Takovýto základní zápis vypadá následovně:

2001:0DB8:0000:0000:0202:B3FF:FE1E:8329

Již na první pohled je vidět, že zápis IPv6 adresy není nikterak jednoduchý. Proto je několik způsobů zjednodušení zápisu. Například každý dvoubytový blok adresy, který obsahuje samé nuly, lze zapsat pouze jako jednu nulu.

Pokud je za sebou několik 2bytových bloků samých nul, lze je zkrátit na pouhé dvě za sebou jdoucí dvojtečky. Toto lze použít pouze jednou v celé adrese, jinak by nebylo možné sestavit adresu zpět do plného tvaru. Takto zjednodušená výše uvedená adresa by mohla vypadat například takto:

2001:DB8::202:B3FF:FE1E:8329

2.2.4 Prefix adresy a jeho zápis

Prefix adresy určuje počet počátečních bitů, které jsou použity pro určení podsítě. Délka prefixu může být různá, záleží zde na úhlu pohledu. Pro někoho může být zajímavý prefix poskytovatele připojení, který bude pravděpodobně kratší než prefix nejaké lokální podsítě [3]. Tento zápis je velice jednoduchý a téměř se nezměnil od dob IPv4, tedy adresa/počet bitů prefixu, např:

2001:15c0:665b::/48

Pro zajímavost zde uvedu několik nejvýznačnějších prefixů [3]:

Zápis	Popis
2000::/3	globální unicastový prefix
FE80::/10	linkový lokální prefix
FF00::/8	multicastová adresa
::/128	nedefinovaná adresa
::1/128	lokální smyčka

2.2.5 Globální individuální adresa

Tata adresa je zřejmě nejdůležitější adresa protokolu IP. Je identifikována binárním prefixem 001. Její formát je definován v RFC 4291 viz obr.2.2. Je potřeba říci, že

Obrázek 2.2: globální routovatelná adresa

rozhraní mezi globálním prefixem a prefixem podsítě není nijak upraveno normou. Podstatný je jen prefix 001/3, doposud jediný pro globální individuální IPv6 adresy.

Poslední část, identifikátor zařízení, je identifikátorem koncového rozhraní a musí být jedinečný na dané podsíti. Z toho vyplívá, při jednoznačnosti prefixu, jednoznačnost celé adresy v internetu. Tento identifikátor je odvozen od z IEEE EUI-64. Je to standart zaměřující se na přidělování celosvětově jednoznačných identifikátorů. Délka těchto identifikátorů je 64 bitů a tudíž odpovídá délce identifikátoru zařízení v IPv6 adrese [3]. Pro případ ethernetu se EUI-64 vytváří z MAC adresy síťového zařízení. Samotná MAC adresa má pouhých 48 bitů, proto se mezi třetí a čtvrtý byte MAC adresy vkládá 16bitový blok nesoucí hodnotu FFFE hexadecimálně [3].

2.2.6 Speciální adresy

V IPv6 se vyskytuje několik speciálních adres. Jedna z nich je např. nedefinovaná adresa [3]

0:0:0:0:0:0:0:0, nebo též ::

Tato adresa se dá použít např. jako zdrojová adresa při odesílání požadavku na automatickou konfiguraci. Zmíněná adresa má též mnoho omezení. Nemůže být přiřazena koncovému zařízení a nemůže být nikdy specifikována jako cílová adresa.

Další speciální adresa je hojně používaná i v nynějším protokolu. Jde o tzv. loopback, v šestkové verzi IP zapisované jako:

0:0:0:0:0:0:0:1, nebo též ::1

Tato adresa je vhodná a používá se tak i v současném protokolu pro testování IP, protože pakety se posílají do IP stacku přímo a to bez posílání do venkovní podsítě. Zmíněná adresa se také používá při komunikaci mezi klientem a serverem, pokud se nacházejí na stejném počítači.

2.2.7 Linkové lokální a lokální IPv6 adresy

V IPv4 jsou poměrně často používány lokální adresy, někdy nazývané neveřejné. Datagramy nesoucí tyto adresy nemohou být routovány do internetu. Toto se významně dotýká problematiky okolo NAT.

I IPv6 disponuje adresami s podobnými vlastnostmi. Jsou to právě linkové lokální a lokální adresy. Původní IPv6 definovalo 2 druhy lokálních adres - lokální linkové a lokální místní, každá s jiným dosahem, rozlišeným prefixy.

Bohužel okolo problematiky lokálních místních adres bylo mnoho nejasností, zvláště kde vůbec končí pole jejich působnosti. Proto byly lokální místní adresy v dalších návrzích protokolu IPv6 označeny za zastaralé a je velice moudré je dále nepoužívat.

Místo nich byly v RFC 4193 zavedeny podobné adresy nesoucí název unikátní lokální individuální adresy, zkráceně lokální IPv6 adresy. Tyto adresy byly zavedeny pro použití uvnitř sítí a podsítí. Lokální adresa by se dala charakterizovat asi takto:

1. Má unikátní globální prefix, který by měl být filtrován na hranicích sítě
2. Je nezávislá na poskytovateli připojení
3. Může být použita pro vnitřní komunikaci bez přístupu k internetu
4. Aplikace s ní mohou pracovat stejně jako s jakoukoliv jinou globální adresou
5. Při nechtěném přesměrování do internetu nehrozí konflikt adres kvůli jejich jednoznačnosti.

Linková lokální adresa je uvozená 10 bitovým prefixem FE80, za nímž následuje 54 nulových bitů a posléze klasický 64bitový identifikátor zařízení.

2.2.8 Anycastová (výběrová) adresa

Výběrové adresy jsou v oblasti adresování poměrně zajímavou novinkou, která ovšem přináší některá nebezpečí. Problém například vyvstává při volbě příjemce. Odešleme-li datagram na výběrovou adresu, dorazí pouze jedinému příjemci, ale odešleme-li sérii datagramů, případně datagram fragmentovaný, může se stát, že každá část dorazí jinému příjemci ze skupiny této výběrové adresy [4]. Výběrové adresy nemají zvláštní prefix, proto nejsou rozpoznatelné od běžných globálních adres.

2.2.9 Multicastová (skupinová) adresa

Skupinové adresy jsou již v současném internetu hojně používány. Implementace skupinových adres v IPv6 přináší změnu tím, že je zde definován dosah skupinové adresy. Dosahem je myšleno, jak daleko od sebe můžou být jednotliví členové skupiny[3].

Identifikátor skupiny zabírá velkou část adresy a to celých 112 bitů. Skupinové adresy se odlišují od ostatních adres 8 bitovým prefixem FF.

Identifikátor skupiny je platný neustále, pokud se jedná o permanentní skupinu a nezávisí na dosahu jednotlivých adres. Jako velmi dobrý příklad poslouží skupinový identifikátor 0x101 hexadecimálně, který je přidělen NTP serverům [3].

FF01:0:0:0:0:0:0:101 - tato skupina sdružuje všechny NTP servery na stejném interface (např. síťové kartě).

FF02:0:0:0:0:0:0:101 - tato skupina sdružuje NTP servery umístěné na stejné síťové lince

FF05:0:0:0:0:0:0:101 - tato skupina sdružuje všechny NTP servery v rámci sítě

FF0E:0:0:0:0:0:0:101 - poslední skupina, největší dosahem, sdružuje všechny NTP servery v internetu.

Poměrně novou věcí v oblasti skupinových adres jsou takzvané dynamicky přidělované skupinové adresy. V tomto případě se jedná hlavně o to, aby adresy byly celosvětově jedinečné a to bez složitého ověřování, zda není tato adresa již někde používána. Ve stručnosti lze uvést, že tento mechanismus přidává do identifikátoru skupiny prefix podsítě, jež by měl být celosvětově jednoznačný, čímž nabývá na jednoznačnosti i skupinová adresa.

2.2.10 Povinné adresy

Každé zařízení připojené pomocí IPv6 musí splňovat určité požadavky. Jeden z požadavků jsou povinné adresy, ke kterým se dané zařízení musí hlásit. Obyčejný počítač, případně jiné zařízení podobné funkčnosti, se musí hlásit k následujícím adresám [3]:

1. lokální linková adresa pro každý interface
2. přiřazená individuální případně výběrová adresa
3. loopback adresa
4. skupinová adresa pro všechny uzly
5. skupinová adresa pro vyzývaný uzel pro všechny přidělené individuální a výběrové adresy
6. všechny skupinové adresy do jejichž skupin patří

Směřovač se navíc musí hlásit k adresám [3]:

1. výběrové adresy podsítě pro všechny podsítě jejichž je směřovačem
2. všechny přidělené výběrové adresy
3. skupinové adresy pro všechny směřovače
4. všechny skupinové adresy jejichž je směřovač členem.

2.3 ICMPv6

ICMP, plným názvem Internet Control Message Protocol slouží k podávání informací o chybách a stavu sítě [4]. Tento protokol samozřejmě existuje i v současném světe IPv4. V IPv6 je ale tento protokol mnohem rozvinutější a slouží nejen k hlášení stavu sítě, ale i k mnohým dalším funkcím.

2.3.1 Druh zpráv

ICMPv6 definuje dva druhy zpráv - chybové a informativní. Mezi chybové zprávy patří Destination unreachable (cíl nedostupný), Packet too Big (příliš velký paket), Time exceeded (vypršení časového limitu) a Parameter problem (chybný datagram). Z informativních to jsou Echo Reply a Echo request, známé jako ping a pong.

2.3.2 Chybové zprávy

Jak již bylo řečeno, existují základní čtyři druhy chybových zpráv, přičemž každá nese ještě podrobnější informace o problému, jako například `No route to destination`.

2.3.3 Informační zprávy

Mezi informační zprávy patří Echo Request a Echo Reply, které se používají k testování IP sítí. Tyto zprávy mají ve svém těle uložen identifikátor a pořadové číslo zprávy. Volání pingu započne sekvenci žádostí, které mají stejný identifikátor, ale pořadové

číslo narůstá, aby bylo možno identifikovat příchozí odpověd [3], zda-li se nějaký paket neztratil nebo například nepřišel zpět později než odpověd na následující paket.

2.3.4 Objevování sousedů

Objevování sousedů bylo navrženo do IPv6 jak náhrada stávajícího mechanismu ARP (Address Resolution Protocol). Ten má za úkol zjistit ethernetovou (MAC) adresu určitého počítače, jehož IP je mu známa. Při vytváření obdobného nástroje pro protokol IPv6 mu jeho tvůrci vnukli spoustu dalších možností a výsledek nazvali Objevování sousedů (Neighbor Discovery). Tento nástroj nyní slouží k následujícím účelům:

1. Zjišťování linkových adres zařízení ve stejné síti
2. Aktualizace neplatných položek a zjišťování změn v linkových adresách
3. Objevování směřovačů, kteří umí směřovat jejich pakety
4. Detekce duplicity IP adres
5. Zjišťování síťových prefixů, routovacích cest a dalších konfiguračních informací

2.3.5 Hledání linkových adres

V tomto ohledu se ND velmi podobá mechanismu ARP pocházejícímu z IPv4. Lze zde nalézt poměrně zásadní změnu a tou je adresa, na kterou se posílá dotaz. V IPv4 se dotaz na linkovou adresu posílá na 255.255.255.255, tedy všem účastníkům sítě, přičemž odpovídá ten, jehož adresa souhlasí s hledanou linkovou adresou.

Pro potřeby hledání sousedů byla definováno několik skupinových adres se společným prefixem FF02:0:0:0:0:1:FF00:/104 [3]. Pro odeslání dotazu na linkovou adresu se k tomuto prefixu připojí posledních 24 bitů z IP adresy a na takto získanou adresu se odešle dotaz. Takováto adresa má speciální název: adresa pro vyzývaný uzel, anglicky solicited node address.

Z tohoto mechanismu a mechanismu tvorby IPv6 adres vyplívá, že v této skupinové adrese již nejsou obsaženy všechny zařízení na síti jako v dobách IPv4, ale jejich počet

bude menší, s největší pravděpodobností bude i na velkých sítích právě jedna adresa, ona hledaná [3]. Také z toho vyplívá, že i když bude zařízení v několika podsítích s ruznými prefixy, adresu pro vyzývaný uzel bude mít stále stejnou na všech podsítích [3].

2.3.6 Detekce dosažitelnosti souseda

Při aktivní komunikaci se sousedy by měl IP dostávat informace o dosažitelnosti souseda od vyšších vrstev. Není-li tomu z různých příčin právě tak, dokáže si IP poradit sám a to sice výše zmíněnou detekcí dosažitelnosti souseda. Proto se v cache sousedů u každé položky vyskytuje několik příznaků, např. nekompletní. dosažitelný, prošlý, odložený a testovaný [3].

2.3.7 Autokonfigurace

Mechanismus autokonfigurace by měl přinést úsporu práce systémovým administrátorům a také jednodušší správu domácích sítí v dobách, kdy bude do domácí sítě mimo PC patřit například televize, chladnička a další podobné zařízení a nikdo nebude potřebovat na své síti běžící DHCP server, což je o další krok blíže k normálním uživatelům. Nutno podotknout, že i v IPv6 je možno používat stavovou konfiguraci, tedy obdobu dnešního DHCP.

Ovšem významné zlepšení je bezestavová konfigurace, kdy se zařízení může nakonfigurovat samo, případně s použitím informací z routeru (samozřejmě bez potřeby DHCP serveru). Pro vygenerování IPv6 adresy použije host některé informace od sebe (např. MAC adresu, případně náhodný identifikátor) a některé informace z routeru na síti.

Ohlášení směřovače je způsob, kterým se dostanou potřebné informace z routeru k hostu na síti. Každý směřovač vysílá v náhodném časovém intervalu své ohlášení a vysílá je na všechny sítě ke kterým je připojen. Tohoto lze využít k jednoduchému přečíslování sítě, protože stačí změnit informaci o prefixu na routeru, který posléze bude distribuovat do sítě [3]. Pokud není na síti žádný router, může si host nakonfigurovat

alespoň lokální adresu s prefixem FE80, pomocí které může komunikovat po lokální linkové síti.

2.3.8 Path MTU discovery

Tento mechanismus v doposud vládnoucím protokolu IPv4 existuje, ale není povinný, jedná se spíše o optimalizační mechanismus. V IPv6 je path MTU discovery povinné kvůli odlišné funkci této vlastnosti. U IPv4 může být paket fragmentován odesílatelem nebo jakýmkoliv routerem po cestě paketu, pokud je na následující trase maximální možná velikost datagramu menší než datagram aktuálně je, kdežto v IPv6 může být paket fragmentován pouze odesílatelem dat.

A právě proto, aby IPv6 poznal, zda-li má paket fragmentovat, je zde zaveden mechanismus path MTU discovery, neboli přeloženo, hledání nejužšího místa na trase z hlediska velikosti MTU (maximum transfer unit). Zjednodušeně tento mechanismus funguje tak, že odesílatel odešle paket s velikostí odpovídající MTU dalšího hopu. Nyní čeká, jestli paket dorazí do cíle a nebo jestli ho nějaký router po cestě zahodí a vrátí chybu "packet too big" [4]. Pokud dojde k chybě, odesílatel nastaví MTU na velikost odpovídající možnostem routeru jenž vrátil chybu a zkusí odeslat paket znovu. Toto opakuje do doby, než se podaří paket doručit do cíle. Paket ovšem nelze zmenšovat donekonečna, proto je určeno jako nejnižší velikost MTU v IPv6 1280 bytů.

2.4 Bezpečnost v IPv6

Ve specifikaci IPv4 nebylo při návrhu protokolu vůbec myšleno na zabezpečení. Proč tomu bylo je poměrně jednoduché. V prvopočátcích sítí byly do internetu připojeny jen úzké okruhy lidí od kterých nevzcházelo příliš velké nebezpečí. Ani výpočetní výkon tehdejších počítačů nebyl největší a každý bezpečnostní mechanismus na výkonu jen ubíral. Na druhou stranu je třeba podotknout, že tyto sítě byly využívány i vojenskými institucemi, kde bylo nezabezpečení na povážnou.

2.4.1 IPSec

Postupem času se IPv4 dočkalo zabezpečení zvané IPSec, jehož implementace není u IPv4 stacku povinná. To se ovšem s IPv6 mění, neboť v protokolu nové generace je přítomnost IPSec povinná [3].

IPSec zabezpečuje autentifikaci a šifrování dat. Při použití šifrování je znemožněno čtení dat tekoucích sítí, protože jsou zašifrována. Takto zašifrovaná data umí přečíst pouze příjemce těchto dat. Autentifikace jednak znamená, že je ověřena skutečná identita počítače na druhé straně a lze tak předejít situaci, kdy se nějaký záškodník vydává za někoho jiného, ve snaze získat nějaká zajímavá data. Autentifikace ale také zabraňuje pozměnění dat po cestě.

Základní části IPSec-u jsou takovéto [3]:

- ESP (Encapsulating Security Payload) zabezpečuje šifrování dat
- AH (Authentication Header) zajišťuje autentifikaci

Tyto rozšiřující hlavičky (ESP, AH) mohou být doplňovány do paketu dvěma způsoby. Tyto způsoby se nazývají transportní nebo tunelující režim. Při transportním režimu se přidává ESP/AH hlavička coby rozšiřující hlavička IPv6, kdežto v tunelujícím režimu se celý stávající paket i s původní hlavičkou zabalí do nového paketu s novou hlavičkou včetně rozšiřující hlavičky ESP/AH.

2.5 Mobilita

V této práci bude poslední teoretickou kapitolou podpora mobility v IPv6. Troufám si tvrdit, že jak je tato vlastnost revoluční, tak stejnou měrou je složitá její implementace. Snad proto bylo zatím konečné RFC týkající se této vlastnosti IPv6 vydáno až v polovině roku 2004, jde stále tedy o vlastnost velice čerstvou, téměř neprobádanou.

Jednoduše popsáno, podpora mobility znamená možnost mobilního uzlu pohybovat se mezi více sítěmi beze ztráty připojení ke své domácí síti, kde má přidělenu tzv. domácí adresu, pod kterou je uzel zanesen v DNS. Tato domácí adresa je globání unicastová adresa a na ní se doručují pakety pro toto konkrétní zařízení [3].

Když se uzel, tedy zařízení, připojí do jiné než do své domácí sítě, ustaví si ve své domácí síti tzv. domácího agenta, který přijímá pakety směřující na domácí adresu zařízení a předává je samotnému zařízení, tou dobou připojeného do jiné sítě, tunelem [3].

Protože takovéto přeposílání dat není ideální, pokouší se mobilní zařízení nacházející se mimo svou domácí síť seznámit původního odesílatele paketů se svou nynější dočasnou adresou. To se nazývá optimalizace cesty. Tomuto musí samozřejmě předcházet ověření mobilního uzlu, aby bylo jasné, že se jedná o to zařízení za něž se vydává.

Pokud se toto vše podaří, začne původní odesílatel opatřovat pakety hlavičkami směřování, které jsou primárně určeny pro domácí adresu uzlu, ale posílány jsou na aktuální dočasnou adresu. Z pohledu protokolů vyšších vrstev je mobilita naprosto transparentní.

Kapitola 3

IPv6 a Linux

3.1 Historický vývoj IPv6 pro Linux

První kód týkající se IPv6 byl přidán do linuxového jádra verze 2.1.8 v listopadu roku 1996 panem Pedro Roquem. V následujícím období nebylo příliš mnoho vývojářů zajímající se o tuto problematiku a vývoj v IPv6 v linuxovém jádře poněkud zaostával. Kolem října roku 2000 vznikl v Japonsku projekt USAGI (UniverSAl playGround for Ipv6), který se snažil implementovat do linuxového jádra chybějící nebo opravit zastaralou podporu IPv6. Mimo to se snaží o vytvoření IPv6 rozhraní (API) v knihovně glibc a upravení aplikací do IPv6 kompatibilní "podoby".

Jejich snažení vycházelo z úzké spolupráce s podobným projektem KAME, který se snažil vytvořit kvalitní podporu pro IPv6 v klonech BSD a projekty WIDE a TAHI. Výsledky práce USAGI spočívaly ve vydávání patchů proti aktuálním zdrojovým kódům linuxového jádra.

Se začátkem vývojové řady jádra 2.5.x se USAGI snažilo dostat jejich kód přímo do zdrojových kódů jádra, protože patche pro stabilní řadu jader 2.4.x byly až příliš veliké a začlenění do jádra bylo problematické. Později byly některé věci do jader řady 2.4.x backportovány, ale bohužel ne všechny nejaktuálnější implementace, což může znamenat problémy při komunikaci s jinými OS.

Naštěstí se v současné době upouští od používání jader z řady 2.4. Pokud jsou tato

jádra někde používána, bývají to stroje buď zastaralé, vykonávající "monotónní" práci bez potřeby nejnovějších implementací IPv6 stacku nebo se tyto jádra nacházejí na vysoce stabilních produkčních serverech, jejichž administrátoři musí oželet nejmodernější technologie na úkor stability.

V současné době vyvíjí USAGI implementaci IPv6 na svém repozitáři zdrojových kódů jádra odvozených od vanilla jádra. Posléze, je-li implementace konkrétní vlastnosti v USAGI repozitáři hotová a otestovaná, integruje se do hlavní větve linuxového jádra.

3.2 Aktuální stav IPv6 v Linuxu

Aktuálním stavem implementace IPv6 dostupné v linuxovém jádře je myšlena implementace IPv6 v aktuálním linuxovém jádře, čímž je nyní verze 2.6.20.1. Dá se předpokládat, že tato a následující verze budou v brzké době základem významné většiny distribucí (např. Fedora, SuSE, Ubuntu apod.). Je také nutné poznamenat, že vývoj v této oblasti jde neustále kupředu a každým dnem se implementace IPv6 opravuje či doplňuje, což je vidět např. na repozitářích projektu USAGI, kde jsou změny v kódu prováděny velmi často, mnohdy denně.

3.2.1 IPv6 Ready

Linuxové jádro je v současnosti držitelem osvědčení IPv6 Ready, konkrétně testů Phase 1 a Phase 2. IPv6 Ready jsou testy součinnosti IPv6 zařízení a shody implementace IPv6 v tomto zařízení s platnými RFC. Každá fáze má několik cílových skupin, ale protože linuxové jádro absolvovalo testy ve skupinách Router a Host, zabývejme se jen jimi.

V každé této skupině se testují ty vlastnosti IPv6, které jsou pro danou roli nezbytná. Testy jsou to velice sofistikované. Pro ilustraci uvedu příklad několika testů Phase 1 pro roli Hosta:

Sekce 3: Stateless Address Autoconfiguration NUT receives DAD NS (target !=

NUT) - testovaný host přijme zprávu o duplicitě adresy, ovšem s nepatřičnou cílovou adresou.

Sekce 5: ICMPv6 Unknown Informational Message Type - testovanému hostu je zaslána nedefinovaná ICMP zpráva

Při těchto testech se zjišťuje, co testovaná implementace IPv6 udělá a zda je to v souladu s doporučením RFC. Aby testovaný host prošel těmito testy úspěšně, musí mít 100% úspěšnost ve všech dílčích testech.

Phase 2 je podobná, jen je narozdíl od Phase 1 směřována více na praktické využití IPv6 implementace.

3.2.2 Možnosti IPv6 v jádře

Linuxové jádro je šířeno pod licenční politikou označovanou jako GNU/GPL, která umožňuje distribuovat jádro OS včetně zdrojových kódů. Této skutečnosti se významně využívá a mnoho zkušenějších uživatelů používá jádra zkompilovaná na míru svého systému a potřebám.

Konfigurace jádra nabízí velké množství nejrůznějších voleb a nastavení všeho co s jádrem operačního systému souvisí, IPv6 nevyjímaje. Stupňů podpory IPv6 v jádře může být několik, od úplné stávající podpory až po úplné vypnutí podpory IPv6 v jádře, případně mnoho stupňů mezi těmito extrémy. Volby jádra v oblasti IPv6 ilustruje obrázek 3.1

Jak je z obrázku vidět, některé možnosti nepovažují vývojáři dosud za příliš stabilní a označují jejich implementaci jako EXPERIMENTAL, čímž varují před používáním této implementace na důležitých místech, kde se počítá s co největší stabilitou. Pro domácí použití by tento experimentální kód příliš vadit neměl. Toto se například týká implementace Mobility v IPv6, která je stále v intenzivním vývoji.

Pro zajímavost uvedu, že volby v této ilustraci označené symbolem * jsou napevno zakompilovány do jádra, kdežto volby označené písmenem M se přeloží jako moduly a lze je do jádra nahrávat a odebírat kdykoliv je potřeba.

```
<*>   The IPv6 protocol
[*]     IPv6: Privacy Extensions support
[*]     IPv6: Router Preference (RFC 4191) support
[*]       IPv6: Route Information (RFC 4191) support (EXPERIMENTAL)
<M>     IPv6: AH transformation
<M>     IPv6: ESP transformation
<M>     IPv6: IPComp transformation
[*]     IPv6: Mobility (EXPERIMENTAL)
<M>   IPv6: IPsec transport mode
<M>   IPv6: IPsec tunnel mode
<M>   IPv6: IPsec BEET mode
<M>   IPv6: MIPv6 route optimization mode (EXPERIMENTAL)
<M>   IPv6: IPv6-in-IPv4 tunnel (SIT driver)
<M>   IPv6: IPv6-in-IPv6 tunnel
[*]   IPv6: Multiple Routing Tables
[*]     IPv6: source address based routing
```

Obrázek 3.1: konfigurace jádra

- První volba, The IPv6 protocol, zapíná obecnou základní podporu IPv6 v jádře, pričemž zůstává funkční i podpora IPv4. Toto bývá označováno jako dual stack.
- Další volba, Privacy extension support, zajišťuje přirazení periodicky se měnící pseudonáhodně generované globální unicastové adresy sítovému rozhraní při použití bezestavové autokonfigurace.
- Volba Router preference je volitelné rozšíření vlastnosti Ohlášení směřovače, podle které se může host lépe rozhodnout, ke kterému routeru se má připojit, vhodné zvláště v tzv. multihomed sítích.
- Volba Route information se váže k předchozí volbě a specifikuje sítové prefixy, které jsou dostupné na routeru. Tato volba je zatím označena jako experimentální.
- Následující tři volby se vztahují k zabezpečení toku paketů pomocí IPSec. V jednotlivých volbách lze povolit různé možnosti tohoto zabezpečení, konkrétně AH, ESP a IPComp, z nichž jsou některé popsány výše v kapitole IPSec.
- Zde mírně přeskočím řazení vlastností IPv6 jádře a zmíním volby IPSec transport mode, IPSec tunnel mode a IPSec BEET mode, povolující tyto módy IPSec přenosu.

- Volba Mobility, dosud vydávaná za experimentální, zapíná tuto podporu Mobility v jádře. Podrobněji o mobilitě pojednává předchozí kapitola.
- Na mobilitu volně navazuje možnost MIPv6 route optimization mode zapínající tuto volbu Mobility, též popsanou výše jako optimalizace cesty.
- Následují dvě volby tunelování v IPv6, a to sice podpora pro tunel IPv6 v IPv4 a tunel IPv6 v IPv6. Tunelování je v podstatě zapoudření jednoho protokolu do druhého.
- Multiple routing table je volba, po jejímž zapnutí je možno využívat více routovacích tabulek, což se využívá např. při pokročilejším routování.

3.3 Ukázková konfigurace IPv6 v Linuxu

Tato ukázková konfigurace bude vycházet z poměrně široce oblíbené distribuce Fedora Core 6 vyvíjené s pomocí společnosti RedHat, jedním z průkopníků Linuxu.

Fedora Core 6 má ve svém jádře zakompilovanou podporu pro IPv6, tudíž pro použití na stanici postačuje dodávané distibuční jádro,v případě potřeby novějšího jádra lze provést jeho update, ve Fedoře konkrétně touto posloupností příkazů:

- `$ su -` Čímž se po zadání rootovského hesla přepneme do superuživatelského režimu potřebného k instalaci jakéhokoliv balíku.
- `$ yum update kernel` - Yum je RedHatí nástroj pro správu balíků a tento konkrétní příkaz vyhledá aktualizace instalovaného jádra a nabídne je k instalaci, což se provede potvrzením dotazu. Po instalaci je nové jádro připraveno k použití a je správně nastaven zavadeč linuxu s implicitní volbou právě nainstalovaného jádra. Nové jádro by se mělo načíst po restartu systému.

Pokud používáme jiné jádro či distribuci, můžeme ověřit dostupnost IPv6 tak, že ověříme existenci tohoto souboru v systému: `/proc/net/if_inet6` .

Pokud není tento soubor v systému, ještě to neznamená definitivní nezdar. Podpora IPv6 nemusí být do jádra zakompilována napevno, ale může být zkompilována jako

modul. V tom případě patřičný modul zavedeme příkazem `modprobe ipv6` . Toto je nutné provést s právy superuživatele.

Pokud je tento modul v systému, ale nenatahuje se automaticky při startu OS, lze doplnit do souboru `/etc/modprobe.conf`, případně `/etc/modules.conf` řádku `alias net-pf-10 ipv6`, která toto zajistí.

Dále jsou potřeba k úspěšnému nasazení systémové utility podporující IPv6. Těch je dnes naštěstí většina. Zřejmě nejzásadnější je balík programů nazvaný iproute2, bez kterého by bylo byť jen přiřazení IPv6 adresy síťovému zařízení velmi obtížné, ne-li nemožné.

Iproute2 podporuje IPv6 od verze 2.4.7, přičemž současná Fedora Core 6 obsahuje tento balík již ve verzi 2.6.16-6, není se třeba obávat nekompatibility většiny programového vybavení, pokud se nezvažuje použití nějakého speciálního programového vybavení.

Pokud opravdu dojde k situaci, že daný program neumožňuje plnohodnotnou spolupráci s IPv6, velmi často za něj lze nalézt nějakou náhradu, která umí vykonávat prakticky stejnou činnost.

Je nutné podotknout, že programy jenž jsou důvěrně známé a používají se v současných IPv4 sítích mívají pro IPv6 funkcionalitu odlišný název. Například známý `ping` má svůj IPv6 protějšek známý pod jménem `ping6`. Velice podobně jsou pojmenovávány některé další utility, namátkou jmenované: `traceroute6, tracepath6`.

Další skupinou jsou programy, jenž jsou při použití IPv6 k uživateli naprosto transparentní a netřeba je nijak upozorňovat na použití IPv6. Zástupcem této sorty jsou například utility `ssh`, `wget` nebo `tcpdump`.

3.3.1 Připojení pomocí tunelu

Jeden z hojně používaných způsobů připojení zařízení k IPv6 síti je takzvané 6in4, kdy se IPv6 pakety balí do normálních IPv4 paketů a posílají tunelem k poskytovateli, který podporuje nativní IPv6. Jeden z takovýchto tzv. brokerů je např. SixXS, sídlící na www.sixxs.net. Tento způsob připojení se používá v případě nedostupnosti nativní

IPv6 konektivity na straně klienta, což je v ČR většina případů.

Následující postup se vztahuje ke konkrétnímu IPv6 brokerovi SixXS v konfiguraci se statickou veřejnou IPv4 adresou a navazujícím statickým tunelem.

Ruční konfigurace tunelu 6in4

Nejprve je potřeba zažádat o přidělení tunelu na některém ze serverů spolupracujícím s SixXS. V tomto konkrétním případě se jedná o server ve Slovinsku, ve městě Maribor. Jednou z podmínek přidělení tunelu je poměrně spolehlivá konektivita ověřovaná pingem, přičemž broker požaduje odezvu pod 100ms.

Po přidělení koncového bodu tunelu je potřeba tunel aktivovat pomocí webového rozhraní a následně ho nakonfigurovat mezi koncovými body, přidělit IPv6 adresu a nastavit routovací záznam na lokálním PC.

Tyto kroky vypadají ve formě příkazů pro Fedora Core 6 (a pro většinu distribucí) takto:

- `ip tunnel add sixxs mode sit local 217.75.209.107 remote 212.18.63.73` Tento příkaz pomocí utility `ip` vytvoří síťové zařízení pojmenované jako `sixxs`, kteréžto je v módu `sit`. Tato zkratka znamená Simple Internet Transition.Lze to vysvětlit jako jednoduchý přechod mezi IPv4 a IPv6. Dále je z příkazu vidět lokální IPv4 adresa, tj. adresa žadatele o tunel a remote, vzdálená adresa, tj. adresa serveru, kde končí IPv4 tunel a data z tohoto bodu putují nativní IPv6 konektivitou.

- `ip link set sixxs up` - příkaz ip s těmito parametry takzvaně nahodí zařízení `sixxs` do stavu UP

- `ip link set 1280 dev sixxs` nastaví maximum transfer unit na hodnotu 1280byte, což je minimální hodnota potřebná pro fungování IPv6.

- `ip tunnel change sixxs ttl 64` nastaví time to live na hodnotu 64. Toto je poměrně důležité, protože standartně je TTL IPv4 paketu nastaveno na inherit, což znamená že IPv4 paket bude mít stejné TTL jako IPv6 paket v něm

zabalený. Toto ale nelze použít například při použití tracepath, kdy je v IPv6 paketu nastaveno TTL na 1 a IPv4 paket by se s tímto TTL dostal jen na nejbližší router. Proto musíme nastavit na tunelu takové TTL, aby se IPv4 paket bezpečně dostal až na druhou stranu tunelu.

- `ip -6 addr add 2001:15c0:65ff:d3::2/64 dev sixxs` tímto příkazem přiřadíme tunelu IPv6 adresu, kterou dostaneme přidělenu od brokera.
- `ip -6 route add default via 2001:15c0:65ff:d3::1 dev sixxs` posledním příkazem se nastaví implicitní záznam do routovací tabulky směrem k druhé straně tunelu skrze zařízení sixxs, tj. tunelem.

Po takovémto nastavení by měl výpis `ip addr show` vypadat zhruba takto:

```
sixxs     Link encap:IPv6-in-IPv4
          inet6 addr: 2001:15c0:65ff:d3::2/64 Scope:Global
          inet6 addr: fe80::d94b:d16b/128 Scope:Link
          UP POINTOPOINT RUNNING NOARP  MTU:1280  Metric:1
          RX packets:7242 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7230 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:752832 (735.1 KiB)  TX bytes:896520 (875.5 KiB)
```

`Sixxs` je název samotného síťového zařízení v konkrétním počítači. `Inet6 addr` položka značí IPv6 adresu. V ukázkovém výpisu jsou vidět dvě IPv6 adresy. Jak bylo řečeno výše, zařízení připojené přes IPv6 mívá mnohdy více než jednu adresu. Vždy je to linková lokální adresa, kterou lze poznat podle prefixu `FE80::/10`, ale také podle rozsahu (scope), který je zde označen jako `link`, tedy linková lokální adresa.

Adresa s rozsahem global je globální individuální adresa, tedy ta, přes kterou se lze spojit s daným PC napříč celým internetem, při ideální podpoře IPv6 v internetu.

Poslední důležitější údaj je `MTU`, které má hodnotu `1280 bajtů`, tedy minimum pro IPv6. Další údaje udávají např. množství přijatých a odeslaných dat (`RX bytes, TX bytes`) nebo typ linky, zde zapouzdřené IPv6 v IPv4 (`encap:IPv6-in-IPv`).

Automatická konfigurace tunelu 6in4

Výše uvedený postup zprovoznění tunelu je použitelný pro téměř jakoukoliv distribuci disponující patřičnými utilitami. Mezi nevýhody tohoto postupu patří složitější nastavení a nemožnost automatického nastartování tunelu, pokud si uživatel tento postup nenaskriptuje sám.

Pro usnadnění zprovoznění tunelu disponují distribuce od RedHatu, čili i Fedora Core 6, takzvanými initskripty, které umožňují automatické zprovoznění tunelu po každém startu, případně na vyžádání. V tomto případě je potřeba vytvořit soubor `/etc/sysconfig/network-scripts/ifcfg-sit1`, který bude mít takovýto obsah:

```
DEVICE=sit1
BOOTPROTO=none
ONBOOT=yes
IPV6INIT=yes
IPV6TUNNELIPV4=212.18.63.73
IPV6TUNNELIPV4LOCAL=217.75.209.107
IPV6ADDR=2001:15c0:65ff:d3::2/64
IPV6_MTU=1280
```

Řádka první definuje na jakém rozhraní bude tunel vytvořen. Zařízení sit1 je první zařízení typu sit, které můžeme použít. Boot protocol zde žádný používat nebudeme, viz druhý řádek. Pokud chceme tunel aktivovat automaticky po startu, doplníme k položce `ONBOOT` hodnotu YES, pokud ne, doplníme NO. Na čtvrtém řádku se zapíná podpora IPv6. Položka `IPV6TUNELIPV4` nastavuje IPv4 adresu protější strany tunelu, tedy brokera. Obdobně se na následujícím řádku nastaví IPv4 adresa a na dalším řádku IPv6 adresa naší strany tunelu. Nakonec se nastaví MTU na hodnotu 1280 jako v případě manuální konfigurace tunelu.

Nyní po nakonfigurování můžeme tunel takzvaně nahodit. To lze provést příkazem `ifup sit1`. Po provedení tohoto příkazu by jsme měli dostat identický výpis síťových zařízení jako v případě manuální konfigurace tunelu.

Role IPv4

Jak z výše uvedeného postupu připojení IPv6 skrze tunel vyplynulo, je k tomuto typu připojení nutné mít funkční IPv4 stack. Proto zde bude letmé seznámení s nastavením IPv4 ve Fedora Core Linuxu.

Buď je IPv4 nastavováno přes DHCP, což je poměrně jednoduché nebo trochu složitější manuální nastavení. Vlastnosti síťového zařízení se nastavují v souboru `/etc/sysconfig/network-scripts/ifcfg-eth0` pro zařízení `eth0`. Jméno souboru se mění podle použitého zařízení. Při použití dhcp vypadá konfigurační soubor asi takto:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

Jde pouze o nastavení jména zařízení, o nastavení zařízení při startu počítače a o povolení DHCP na tomto rozhraní.

Pokud se jedná o manuální nastavení parametrů síťového zařízení, vypadá konfigurace následovně:

```
DEVICE=eth0
IPADDR=217.75.209.107
NETMASK=255.255.255.128
ONBOOT=YES
GATEWAY=217.75.209.1
TYPE=Ethernet
```

Zde je již potřeba nastavit IPv4 adresu s příslušnou maskou a bránou. Jinak jsou použity obdobné volby jako v konfiguraci přes DHCP.

Zkušenosti se SixXS

Na základě osobních zkušeností s tímto brokerem mohu říci, že tunel je stabilní a nedochází k jeho rozpadání a také rychlost a odezva IPv6 konektivity poměrně věrně

kopíruje rychlost a odezvy IPv4 konektivity na kterou je v tomto případě fakticky vázána.

3.3.2 Nativní IPv6

Další ze způsobů připojení je připojení do sítě s nativní podporou IPv6. Tuto možnost je možné použít u poměrně malého množství poskytovatelů v ČR. Jedním z nich je např. síť CESNET2.

Tuto možnost jsem vyzkoušel v síti LIANE, počítačové síti technické univerzity v Liberci, která je připojená do právě zmiňovaného CESNETu.

V LIANE je konfigurace IPv6 distribuována ke koncovým počítačům pomocí DHCPv6. V tomto případě stačí přidat do souboru konfigurace síťového zařízení řádku obsahující `DHCPV6C=yes`, čímž je zajištěno poslouchání dhcpv6 klienta pro příslušné rozhraní. Takový konfigurační soubor může poté vypadat takto:

```
DEVICE=eth0
BOOTPROTO=dhcp
DHCPV6C=yes
ONBOOT=yes
```

V tomto konkrétním případě po restartu služby sítě provedené příkazem `service network restart` dostalo síťové zařízení tyto adresy:

```
147.230.158.77
2001:718:1c01:157:20a:e4ff:fe4f:5cef
```

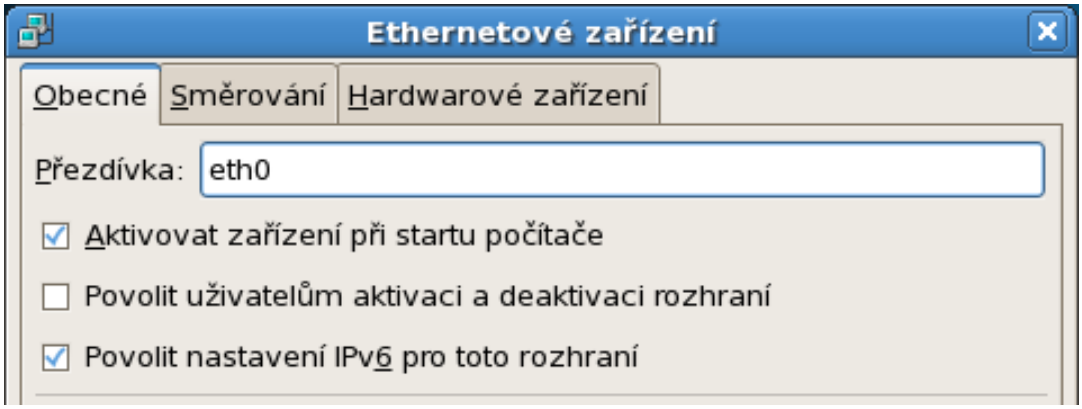
Pokud nechceme trvale používat IPv6 a chceme IPv6 jen vyzkoušet nebo máme jiné důvody proč nezasahovat do initskriptů, pak můžeme využít přímo IPv6 dhcp klienta. Stačí spustit `dhcp6c eth0`, kde místo `eth0` dosadíme zařízení, jimiž jsme do sítě připojeni.

Také můžeme přiřadit IPv6 adresu a příslušný záznam do routovací tabulky sami. Příkaz `ip addr add 2001:718:1c01:157:20a:e4ff:fe4f:5cef dev eth0` přiřadí zařízení eth0 danou adresu a příkaz

`ip -6 route add 2001:718:1c01:157:20a:e4ff:fe4f:5cef/64 via dev eth0` zapíše do routovací tabulky příslušný záznam.

3.3.3 Konfigurace v GUI

Doposud jsme nastavovali parametry sítě v tzv. textovém režimu, ale mnohé distribuce obsahují utility pracující v grafickém režimu, Fedora Core 6 nevyjímaje.



Obrázek 3.2: konfigurace síťové karty

Po instalaci těchto vymožeností přišlo zklamání. Jak je vidět z obrázku 3.2, jediná položka týkající se tohoto protokolu je samotné povolení IPv6 na definovaném síťovém rozhraní. Jakýkoliv pokus zadat do políčka IPv6 adresu skončil bez odezvy. Zadaná adresa se nepřiřadila na síťové zařízení.

Protože jsem byl tímto nepříjemně překvapen, zkusil jsem nainstalovat Fedora 7 test 3, což je beta verze nadcházející verze Fedory 7, u které jsem doufal ve zlepšení v tomto směru. Bohužel, konfigurační dialog zůstal stejný, tedy bez jakékoliv pokročilejší podpory IPv6.

Na druhé straně budou asi největší měrou používány IPv6 autokonfigurace a DHCPv6, takže stávající možnosti jsou pro běžné uživatele dostačující.

3.3.4 Ověření funkčnosti a diagnostika

Může se stát, že přes sebepečlivější nastavení připojení přes IPv6 nebude připojení částečně nebo úplně fungovat. Proto je vhodné znát způsob, kterým lze ověřit funkčnost

sítě, případně diagnostikovat problém.

Jednou z důležitých záležitostí je správné přiřazení adresy k rozhraní a správné záznamy v routovací tabulce. Přiřazené adresy lze vypsat příkazem `ip -6 addr show`. Tento příkaz vypíše všechny síťové zařízení spolu s jim přiřazenými adresami. Pokud nás zajímá pouze jedno síťové zařízení, příkaz se pozmění asi takto: `ip -6 addr show dev eth0`, kde eth0 je jméno zařízení. Routovací tabulku lze vypsat obdobným příkazem `ip -6 route show`.

Pokud je nastavení správné, ale síťové připojení nefunguje, je několik možností jejího otestování. Často používaný nástroj je ping, jehož použití je snadné. Příkazem `ping6` se začne odesílat na zvolený počítač žádost o odpověd. Výstup tohoto testování vypadá následovně:

```
daniel.rimal@sirius:~> ping6 2001:15c0:65ff:d3::2
PING 2001:15c0:65ff:d3::2(2001:15c0:65ff:d3::2) 56 data bytes
64 bytes from 2001:15c0:65ff:d3::2: icmp_seq=1 ttl=52 time=70.6 ms
64 bytes from 2001:15c0:65ff:d3::2: icmp_seq=2 ttl=52 time=86.1 ms
64 bytes from 2001:15c0:65ff:d3::2: icmp_seq=3 ttl=52 time=69.9 ms
--- 2001:15c0:65ff:d3::2 ping statistics ---\
3 packets transmitted, 3 received, 0% packet loss, time 2009ms\
rtt min/avg/max/mdev = 69.937/75.582/86.188/7.507 ms
```

Pomocí příkazu `traceroute6 "cílová adresa"`lze například vypsat cestu, kterou paket projde k cílovému počítači a případně lze takto zjistit, kde tok paketů končí. Výpis traceroute6 vypadá asi takto (zkráceno):

```
[root@karlshorst ~]# traceroute6 www.kame.net
traceroute to www.kame.net (2001:200:0:8002:203:47ff:fea5:3085)
 1  gw-212.mbx-01.si.sixxs.net (2001:15c0:65ff:d3::1)  3
 2  maribor3-vlan-4.amis.net (2001:15c0:ffff:7::1)
 3  M6-Tu6.Space.Net (2001:608:0:3::218f:1)
 4  Cisco-M-XII-Fa0-0.Space.Net (2001:608:0:10::115)  5
 5  Cisco-M-XX-P1-0.Space.Net (2001:608:0:3::15a3:202)
```

Ověřit lze také funkčnost DNS pro IPv6 pomocí dotazu `dig aaaa www.kame.net`, na který by měl DNS server vrátit odpověd obdobnou této:

```
www.kame.net.   86180   IN   AAAA   2001:200:0:8002:203:47ff:fea5:3085
```

Na pokročilejší diagnostiku se používá například `tcpdump`, který zobrazuje pakety které jdou přes definované síťové rozhraní.

3.3.5 Firewall

Zabezpečení počítačů je stále více přetřásaným tématem. S připojením pomocí IPv6 nabývá potřeba zabezpečení stále většího významu, neboť v těchto případech se upouští od jakéhosi krytí počítačů překladem adres a zařízení připojená do sítě jsou víceméně celosvětově dostupná.

V Linuxu se téměř výhradně používá software nazývaný `iptables`, případně `ip6tables`. Obě utility vyvíjí skupina lidí okolo projektu netfilter.org. Ip6tables je program na konfiguraci pravidel paketového filtru.

Pro ilustraci zde uvedu vlastní firewall sestavený z pravidel ip6tables.

```
#!/bin/bash
# Nastaveni promenych skriptu
NET_ADDR="2001:15c0:65ff:d3::2 "
NET_IFACE="sixxs"
LOOP_IFACE="lo"
LOOP_IP="::1/128"
IP6TABLES="/sbin/ip6tables"
#
# Zacatek paketovych pravidel
#
# Nastaveni politiky
$IP6TABLES -P INPUT DROP
$IP6TABLES -P OUTPUT ACCEPT
$IP6TABLES -P FORWARD DROP
```

```
# Povolene porty
$IP6TABLES -A INPUT -i $NET_IFACE -p ipv6-icmp  -j ACCEPT
$IP6TABLES -A INPUT -i $NET_IFACE -p TCP --dport 10002 -j ACCEPT
$IP6TABLES -A INPUT -i $NET_IFACE -p TCP --dport 80 -j ACCEPT
$IP6TABLES -A INPUT -i $LOOP_IFACE -j ACCEPT

# Povoleni navazaneho spojeni
$IP6TABLES -A INPUT -d $NET_ADDR -p tcp ! --syn -j ACCEPT
```

Na začátku skriptu je nastavení proměnných, které se používají dále. Dělá se to z toho důvodu, abychom např. při změně jména síťového zařízení nemuseli měnit celý skript - změní se pouze obsah proměnné NET_IFACE.

Prvním nastavením paketového filtru je nastavení politik pro jednotlivé řetezce pravidel. Řetezce INPUT a FORWARD je nejlépe nastavit na DROP. Poté je každý paket procházející tímto pravidlem zahozen. Pakety jdoucí přes OUTPUT pocházejí z našeho počítače, je tedy možno ponechat bez omezení.

Protože v tuto chvíli se veškerá příchozí komunikace zahazuje, je potřeba povolit takové služby, které potřebujeme mít zvenčí přístupné. Rozhodně je dobré povolit icmp protokol. Dále jsou to porty 10002, což je SSH server, byť na poněkud neobvyklém portu. Další je klasický port 80, tedy web server. Také je vhodné povolit veškerý provoz na loopbacku.

Posledním pravidlem tohoto jednoduchého firewallu je povolení spojení, které nemají nastavený příznak SYN. Tyto spojení nejsou navazující a takový paket pravděpodobně již patří nějakému navázanému spojení. Bez tohoto povolení by na počítači nefungovala žádná internetová komunikace, protože by sice od nás paket odešel, ale pokud by nepřicházel zpět na nějaký povolený port, byl by zahozen a k nám by se nedostal.

3.4 Výkonnost IPv6

Výkonnost síťových operací, stejně jako výkonnost ve výpočetní technice obecně, bývá na prvních místech zájmu té či oné technologie. Proto se pokusím změřit a srovnat rychlost implementace IPv4 a IPv6 stacku v linuxu.

3.4.1 Metoda měření

K tomuto měření jsem použil program `iperf`, který je volně dostupný ke stažení na adrese `http://dast.nlanr.net/Projects/Iperf`. Na téže adrese jsou dostupné i zdrojové kódy, neboť tato utilita je licencována pod GNU/GPL.

Testování spočívalo ve spustění `iperf` v módu server na jednom konci měřené linky a ve spuštění klienta na opačné straně linky. Klient se připojil na zadanou IP adresu serverové části a snažil se "protlačit" skrze síť co největší množství dat. Z přijmutého množství dat a času, po který byla data zasílána, se spočítala propustnost linky.

Rychlost IPv4 a IPv6 stacku jsem porovnal tím, že jednou jsem dvojici server-klient spustil na IPv4 stacku, podruhé, za stejných podmínek, na IPv6 stacku, který iperf též podporuje.

Standartně iperf pracuje s TCP, volitelně s UDP. Při testu byla použita implicitní konfigurace všech voleb programu iperf. Při testu byly síťové karty obou testovacích PC spojeny přímo proti sobě patch kabelem cat 5e a síťové karty byly nastaveny na rychlost 1000Mb/s full duplex.

3.4.2 Použitý hardware

Test probíhal na dvou počítačích s obdobnou konfigurací.

Základní hardware

První počítač byl osazen základní deskou `Supermicro X7DB3`, s 2GB DDR2 ECC Fully Buffered operační paměti a pouze jedním procesorem `INTEL Dual-Core`

Xeon 5130 na taktu 2.0GHz. Diskový subsystém zde tvořilo 9 disků po 500GB, z toho 8 v konfiguraci RAID 6.

Druhý počítač, či spíše server, byl též postaven na základu Supermicro, konkrétně PDSMi+, obsahoval 1GB DDR2 operační paměti a procesor INTEL Dual-Core Xeon 3050 taktovaný na 2.13GHz. Na jednom disku byl nainstalován systém, čtyři další instalované disky byly uspořádány v RAID 5.

Síťové karty

Nejzajímavější z hlediska síťových testů jsou síťové karty. Oba servery měly síťové karty integrovány na základní desce a oba shodně obsahovaly po dvou síťových kartách.

První server měl integrovanou síťovou kartu Intel(ESB2/Gilgal)82563EB. Jedná se o moderní duální gigabitovou serverovou síťovou kartu se spoustou zajímavých vlastností.

Druhý server měl na základní desce integrovány dvě rozdílné gigabitové síťové karty a to sice Intel 82573V a Intel 82573L. Tyto síťové karty se funkčně ničím neliší.

Jednou ze zmiňovaných zajímavých vlastností instalovaných síťových karet je například offloading. To znamená, že síťová karta přebírá část zatížení procesoru a některé části zpracování síťového provozu zpracovává vlastním procesorem. Tyto části se liší podle použité karty.

Všechny síťové karty podporují offloading těchto částí zpracování toku dat: rx-checksumming, tx-checksumming, scatter-gather, tcp segmentation offload.

3.4.3 Použitý software

Na obou serverech byl nainstalován Linux, konkrétně CentOS 5 pro platformu X86_64. První server běžel na vanilla jádře 2.6.20 opatchovanému kvůli podpoře diskového řadiče. Na druhém serveru bylo standartní distribuční jádro verze 2.6.18.

Všechny síťové karty byly obshospodařovány ovladačem, resp. modulem e1000 s implicitními parametry.

Při testech na serverech neběžely kromě nezbytných systémových služeb žádné další služby a žádná běžící služba nevytěžovala významněji jakoukoliv část systému.

3.4.4 Výsledky měření

Jednoduché zapojení

První měření spočívalo ve spojení serverů jednou síťovou kartou na každém z nich a následném měření rychlosti spojení při použití IPv4 a IPv6 stacku.

Konkrétní příkaz pro serverovou, resp. klientskou část, testu vypadal takto:

```
iperf -s (-V) -f M
iperf -c 192.168.1.1 (2001:15c0:65fe:d3::2 -V ) -f M
```

pozn.: volby v závorce se týkají IPv6.

Měření bylo prováděno 5 krát pro IPv4 i pro IPv6. Naměřené hodnoty shrnuje následující tabulka:

IPv4 rychlost [KB/s]	**IPv6 rychlost [KB/s]**
115170	113047
114978	113048
114984	113049
114980	113047
114978	113044
průměr [KB/s]	**průměr [KB/s]**
115018	113047
průměr [MB/s]	**průměr [MB/s]**
112,32	**110,40**

Bonding

Při druhém měření jsem použil technologii známou jako `bonding` nebo také `trunking`. Při bondingu se spojí několik síťových karet do jedné skupiny, která zabezpečuje podle zvoleného módu vyšší propustnost a spolehlivost.

Spojil jsem tedy obě síťové karty na každém serveru do zařízení `bond0` v módu round-robin a síťové karty mezi servery vzájemně propojil. Teoreticky by se měla rychlost přenosu blížit dvojnásobku rychlosti jedné karty. S tímto zapojením jsem provedl stejné měření jako v případě jedné karty.

Bohužel nemám k dispozici přesné rychlosti v KB/s, ale pouze v MB/s. Nemá cenu zde vypisovat tabulku naměřených hodnot, protože všech 5 měření dalo stejný výsledek.

Přenos přes IPv4 dosáhl `224MB/s` a přenos přes IPv6 dosáhl `221MB/s`.

Závěr

Cílem této práce bylo zjištění praktické použitelnosti linuxového operačního systému v prostředí sítí s podporou IPv6. Ačkoliv se tato technologie vyvíjí již řadu let a nabízí řadu vylepšení, není stále běžně dostupná a je pro mnohé velkým neznámem.

Toto by se mohlo poměrně brzy změnit, neboť jak je v této práci pojednáno, podpora IPv6 a doprovodných mechanismů je v Linuxu na velmi vysoké úrovni, ať podporou přímo v jádře Linuxu, tak v serverových i uživatelských aplikacích

Rychlost IPv6 stacku v jádře Linuxu sice zaostává, ale jedná se o velmi malé rozdíly, konkrétně o rozdíl 1,7% ve prospěch IPv4 při měření rychlosti jedné síťové karty. Při použití bondingu rozdíl téměř mizí. Myslím, že tento malý handicap může být přičten zpracování čtyřikrát delší adresy, případně téměř dvojnásobné době vývoje IPv4 stacku.

Směrem do komerční sféry se také dá vysledovat poměrně velký posun. Firma RedHat, velký hráč na poli korporátního Linuxu, oznámil při představování své distribuce RedHat Enterprise Linux 5 kompletní podporu IPv6 v tomto produktu s následným dalším vývojem. A protože firma RedHat je jedním z velkých prispěvovatelů do bohatství Linuxu, dá se v této oblasti čekat ještě další rozvoj.

Literatura

[1] Cisco Systems Inc. *Internet Protocol Version 6 Q&A*
URL: http://www.cisco.com/en/US/products/ps6553
/products_qanda_item0900aecd803715bf.shtml

[2] Secunia Headquarters. *IPTables/Netfilter Denial of Service Vulnerabilities*
URL: http://secunia.com/advisories/9429/

[3] SATRAPA, Pavel. *IPv6*. Neocortex spol. s.r.o., 2002
ISBN 80-86330-10-9

[4] HAGEN, Silvia. *IPv6 Essentials*. Second Edition, May 2006. O'Reilly
ISBN-10: 0-596-10058-2

[5] TAHI Project. *Test Sample Result (Self-test Ver. 1.4.6)*
URL: http://www.tahi.org/ume/results/Self_Test_1-4-6/
freebsd61.host/icmp.p2/index.html

[6] TAHI Project. *Test Sample Result (Self-test Ver. 1.4.6)*
URL: http://www.tahi.org/ume/results/Self_Test_1-4-6/
freebsd61.host/addr.p2/index.html

[7] Cisco System. *IPv4 and IPv6 headers*
URL: http://www.cisco.com/image/jpg/en/us/guest/tech/tk872/c1550/cdccont
_0900aecd8054d37d_0900aecd8054d37d-03.jpg